The Provision of Induction in Data Model Systems: I. Analogy

David Hartzband and Laura Holly
Digital Equipment Corporation
Fred Maryanski*
EECS Department, University of Connecticut

ABSTRACT

Hartzband [1] and Hartzband and Maryanski [2] have proposed that the foundation of knowledge-based systems must be an underlying compositional data model—that is, a data model that provides an object structure (the set of representational primitives supplied), an operator structure, and an inference structure (rules and relationships that control the behavior of the system). To be effective, this data model should be isomorphic with the user's perception of the representation of the information. In addition, these workers have proposed that two functions that must be provided by such a system are representation of complex and/or abstract information as well as data values, and the ability to make nontrivial inferences using this information.

One of the types of inference that is important in providing this perceptual isomorphism is analogy, or the evaluation of similarity among represented objects. This article describes a data model for knowledge-based systems. The inference structure of this model includes a reference structure that represents information for individual object instances. Analogy is computed as the evaluation of similarity among reference structures of object instances. Algorithms and examples are presented for several different variations.

KEYWORDS: ***analogy, induction, inference, knowledge-based, model-theory***

---

*The work of Fred Maryanski was partially supported by grants from the National Science Foundation, ECSW-84014787, and Digital Equipment Corporation.

Address correspondence to David Hartzband, Digital Equipment Corporation, HLO2-3/E09, 77 Reed Road, Hudson, Massachusetts 01749.

International Journal of Approximate Reasoning 1987; 1:5-21

52 Vanderbilt Ave., New York, NY 10017 0888-613X/87/$3.50

1. INTRODUCTION

Webster's *New World Dictionary* [3] defines analogy as (1) similarity in some respects between things otherwise unlike, and (2) the inference that certain admitted resemblances imply probable further similarity. Much work on analogy has been directed toward philosophical, psychological, mathematical, and (more recently) computational use of this inference method. This introduction will briefly review this work and position the current work that is the object of this paper. The development of analogy in philosophy will not be treated.

Psychological theories of analogic reasoning stem mainly from the work of Spearman [4, 5], particularly from his "information-processing" (compare Sternberg [6], p. 106) line of thought. Spearman breaks analogy into three qualitative catagories:

1. Apprehension of experience—that is, the classification and encoding of each item in the analogy.
2. Eduction of relations—that is, the drawing out of similarities among the specified items in the analogy.
3. Eduction of correlates—that is, the use of these relations (similarities) to correlate the items in the analogy with items external to it.

This model corresponds to the following formula:

$$A{:}B{::}C{:}[x_1, x_2, \ldots, x_n]$$

Thus, A is to B as C is to some subset (possibly singular) of x.

Sternberg has two criticisms of this theory. First, there is no mention of the process of discovering the connection between the relations in $A{:}B$ and C; the two sides of the analogy are not related. Second, the theory leaves the process "buried in thought," that is, not communicated. Sternberg [6] proposes a variation of this theory to address these objections. This theory consists of five (optionally six) stages:

1. Encoding: potentially revelant characteristics of the first two terms ($A{:}B$) are identified.
2. Inferring: relations between characteristics of the first two terms are identified.
3. Mapping: the third term (C) is encoded and the relationship of the first two terms and the third term is elucidated ($A{:}B{::}C$).
4. Application: this relationship is applied to the set of alternatives (x) and the final terms are selected, thus resolving the analogy.
5. Response: this resolution is communicated.
6. Justification (optional): error checking is undertaken or additional information is determined.

Many other workers have proposed theories of analogic reasoning, but most are

modifications of Spearman's work (Shalom and Schlesinger [7], Rumelhart and Abrahamson [8]).

Several computational theories of analogy have been developed and implemented as programs. Among the best known are ARGUS (Reitman [9]) and ANALOGY (Evans [10]). ARGUS is a complex system that utilizes a semantic net and an overall executive module that performs planning and parallel-element net searches to solve analogy problems of the type used in psychometric testing. ANALOGY resolves geometric similarity problems of the same type by decomposing the geometric objects in the analogy into a higher-level (or lower-level, depending on your point of view) description that is then used in a manner similar to Sternberg's in order to determine which of five figures resolves the problem. Several extensions to these programs have been suggested (Winston [11], Williams [12]).

Carbonell [13, 14] has described two forms of analogy to be computationally realized as problem-solving methods. These are transformational analogy, which is the recognition and incremental transformation of past solutions in new contexts, and derivational analogy, which is the recognition and utilization of past reasoning processes in new contexts. The ARIES system implements transformational methods, and several systems are under development to implement derivational methods. Silverman [15] has proposed a theory of analogy for systems management that is a variation of the Sternberg theory.

The mathematical work on analogy has still best been elucidated and summarized by Polya [16]. Polya describes analogy with reference to its relation with other forms of inductive reasoning: specialization and generalization. Analogy here is the reduction of similarity among (mathematical) objects to definite concepts or relations. Analogy to Polya is pragmatic, a problem-solving method. In comparison to psychological analogy, it can be represented as follows:

1. A::?
   Given A, what is similar to it?
2. [A, B, . . . ,]::? where [A, B, . . . ,] is disjunctive.
   Given the set of objects [A, B, . . . ,], what is similar to it?

Too much work has been done in this area since Polya's description to be summarized here. It is interesting to note, though, that this form of analogy is similar to that used in recent automatic programming systems and in computational learning systems (Lenat and Brown [17]).

The motivation for the work done on the description of the analogy in psychology appears to be primarily related to the description of specific theories of intelligence and the testing of such theories. Much previous work on computational analogy appears to be an attempt to develop a capability for successfully resolving psychological tests of analogy. In contrast, the mathematical characterization of analogy emphasizes problem solving and discovery

through the recognition of analogic relationships in unique and even nonlogical ways (Polya [16], p. 21).

DATA MODEL DESCRIPTION

Hartzband [1] and Hartzband and Maryanski [2] have proposed that the foundation for knowledge-based systems must be an underlying compositional data model—that is, a data model that provides the following:

1. An object structure, which is the set of representational primitives supplied.
2. An operator structure.
3. An inference structure, which includes rules and relationships that control the behavior of the system.

This model might conceptually resemble Figure 1. Object and operator structure are user visible and are embedded in the inference structure. Inference structure may be reflected in operator structure but is most visibly reflected in system behavior. When a user applies an operator to an object, a specific procedure or set of procedures is carried out that results in the return of an object or set of objects to the user. Such objects might be newly created and/or existing objects that may have been modified by the operator. These procedures (specified by operators) and objects are mediated by the inference structure of the system.

In order to be effective, this data model should be isomorphic with the user's perception of the representation of the information. That is, the expressional model must provide a (close to) one-to-one correspondence of objects at the user-perceptual (user model) and machine-conceptual (data model) levels. (This isomorphism need not apply to the machine-internal level.) Figure 2 shows the relationship of these levels.

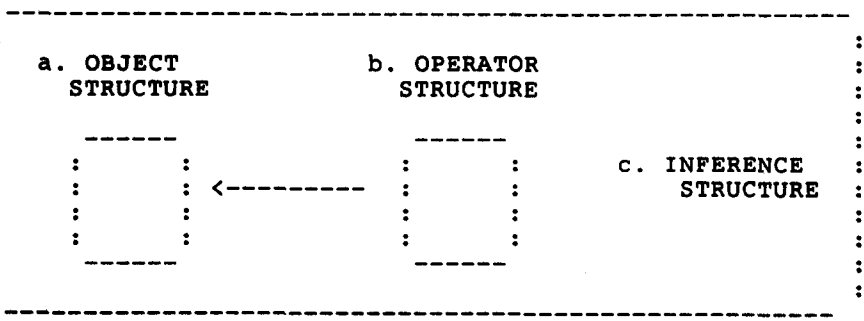

Figure 1. Conceptual Data Model Structure (a. Provides user-visible objects for use in representation of enterprise or problem space; b. Provides user-visible operators for management and manipulation of objects; c. Provides system "behavioral" rules for inference and control of object integrity that may not be user visible.)

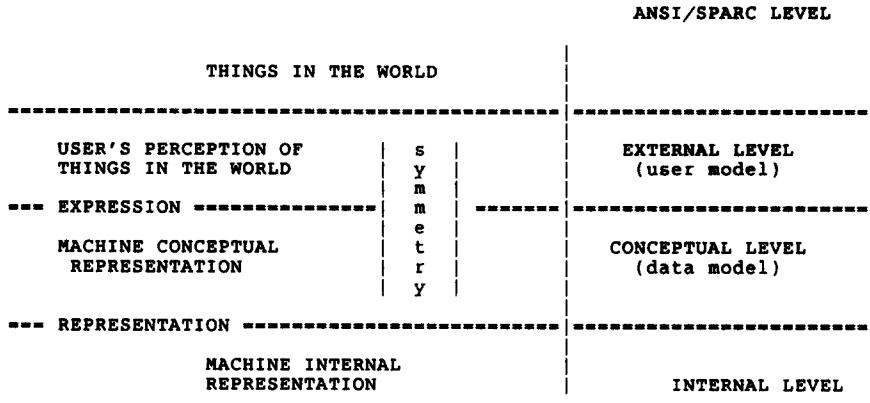

Figure 2. Relationships of Model Levels

There are several advantages to taking a data model approach to provision of (computational) analogy as a problem-solving method. These include integration of inference with the knowledge representation structure provided by the model, and integration of inference, represented in part here by analogy, into the actual model. The integration of inference with the knowledge representation structures of the model is accomplished by a set of algorithms (for provision of inference) that use the structures to compute inferred results. This set of algorithms becomes part of the inference structure of the model that accomplishes the integration of inference with the model itself. The integration is demonstrated by the specification of an object structure, operator structure, and partial inference structure for the data model.

The object structure consists of the following four object types:

1. Entities, or things in the model space (very broadly, nouns).
2. Relationships, or binary directed mappings between object pairs.
3. Sets, or collections (possibly heterogeneous) of objects.
4. Attributes, or descriptors of objects; attribute instances are associated with values.

The operator structure consists of a language expression of operators that corresponds to the types of analogy to be provided (see next section). The partial inference structure consists of two parts: a reference structure for objects, and a set of analogy rules and algorithms. The reference structure has the following form. There exists *Rs*, an object reference structure for each object $x, y, \ldots, n$ of form

$$Rs \langle \text{OBJ\_}x \rangle ::= \langle \text{reference\_list} \rangle$$

$$\langle \text{reference\_list} \rangle ::=$$

$$[\langle \text{ ATR\_}x \ \& \ \text{VAL\_}x \ (\text{numeric}) \ldots . \mid$$

ATR_x & VAL_x (nonnumeric) ⟩ . . . .
⟨ REL_x & OBJ_x ⟩ . . . .
⟨ SET_x ⟩ . . . .
⟨ REL_x ⟩ . . . .
⟨ ATR_x ⟩ . . . .]

That is, each object can be characterized by its attribute/value pairs (numeric and nonnumeric), its relationship/object pairs, its set membership, and its unspecific relationships and attributes. Analogy is defined as the evaluation of an identity relationship ($|I|$) between elements of Rs for specified objects. In fact, this relationship need not be strict identity but can be some more inexact form of similarity. The following rules hold for this relationship:

$$Rs\_x\_\text{element1}(,\ldots,n) \quad |I| \quad Rs\_x\_\text{element1}(,\ldots,n)$$

$$Rs\_y\_\text{element1}(,\ldots,n) \quad |I| \quad Rs\_x\_\text{element1}(,\ldots,n) \quad \neq$$
$$Rs\_x\_\text{element1}(,\ldots,n) \quad |I| \quad Rs\_y\_\text{element1}(,\ldots,n)$$

The reference structures of each object in the analogy are compared, and the similarities among reference structures are returned as the resolution of the analogy. The first rule states that analogy in this manner is reflexive, whereas the second rule states that it is not symmetric; that is, A may not be similar to B in the same way B is similar to A. (This is a consequence of the form of comparison used here—specifically, the use of an object's reference structure as the template for a comparison.) In addition, analogy does not appear to be transitive (it is not necessarily the case that if A is similar to B and B is similar to C, A is similar to C). More correctly, there appear to be cases where analogy (in the form here described) is not transitive; therefore, in general it does not appear to be transitive.

The provision of weights associated with elements of the reference structure can provide an additional capability for resolving analogy in this system. Each element of the reference structure is assigned a weight, and this weight is scored for matches among objects. These weights can be system supplied (as defaults) or user supplied. Furthermore, a capability for statistical matching of numeric attributes can be used to provide "inexact" similarity for these elements.

The actual "measure" of analogy is provided for by the inference structure of the model and the algorithms that implement this structure. In the qualitative case, the measure is simply the list of similarities generated by the comparison. This list is then interpreted by the user as appropriate. In the quantitative case, the measure is both the normalized measure of similarity provided by the algorithm and the similarity list that is generated. The normalized scores are relative to either the proposed weighting structure or to user-supplied weights relevant to a specific problem. In both cases, the interpretation of the similarity list by the user is the real content of the analogy.

PROVISION OF ANALOGY

Several variations of analogy can be provided in this data model system, including but not limited to the following:

1. *A*::? *B*—comparison of two specific objects.
2. Quantitative (weighted) comparison of two objects.
3. *A*::? [*B*, *C*, *D*, . . . ,]—quantitative (weighted) comparison of a specific object with a set of objects.
4. [*A*, *B*, *C*, . . . ,]::? [*W*, *X*, *Y*, . . . ,]—quantitative (weighted) comparison of a set of objects with a set of objects.
5. *A*::?—given a specific object, are any other objects similar?

The first two types of analogy are simply comparisons of the reference structures of two individual objects (*A* and *B*). In the third, a common reference structure is derived for the group [*B*, *C*, *D*, . . . ,] and compared to the reference structure of the individual object *A*. In the fourth, reference structures for both groups are derived and compared; and finally, in the fifth, a set of objects with reference structure similar to the reference structure of *A* is identified (discussed in further detail shortly).

Examples for the two simplest variations are now given. Some workers have suggested that it may be necessary to evaluate dissimilarity as well as similarity for the provision of analogy (Tversky [18]). The algorithms for various cases given in this paper (see the appendix) could be modified to take this into account.

The first example is for a binary-weighted comparison [*A*::? *B*]. The two objects are both instances of the set object type with the following reference structure:

Rs ⟨ OBJ_*A* ⟩ ::= ⟨ reference_list ⟩

⟨ reference_list ⟩ ::=

[⟨ ATR_*x* & VAL_*x* (numeric) . . . . |
ATR_*x* & VAL_*x* (nonnumeric) ⟩ . . . .

project-type administrative	{weight = 1.0}
accounting-code ovd	{weight = 1.0}
⟨ REL_*x* & OBJ_*x* ⟩	
has-cost-center 306	{weight = 0.8}
⟨ SET_*x* ⟩	
information management group	{weight = 1.5}
⟨ REL_*x* ⟩	
has-cost-center	{weight = 0.6}
⟨ ATR_*x* ⟩	
project-type	{weight = 0.5}
accounting-code	{weight = 0.5}]

Rs ⟨ OBJ_*B* ⟩ ::= ⟨ reference_list ⟩

⟨ reference_list ⟩ ::=

[⟨ ATR_*x* & VAL_*x* (numeric) . . . . |
ATR_*x* & VAL_*x* (nonnumeric) ⟩ . . . .

project-type research	{weight = 1.0}
accounting-code res	{weight = 1.0}
⟨ REL_*x* & OBJ_*x* ⟩	
has-cost-center 306	{weight = 0.8}
⟨ SET_*x* ⟩	
information management group	{weight = 1.5}
⟨ REL_*x* ⟩	
has-cost-center	{weight = 0.6}
⟨ ATR_*x* ⟩	
project-type	{weight = 0.5}
accounting-code	{weight = 0.5}]

The algorithm compares each element of the reference structure of Object*A* to the corresponding element in the reference structure of Object*B*. If an exact match is found, the element along with its weight is written to a similarity list. After all elements have been compared, the weights are summed and normalized between 0 and 10 (0 = no similarity and 10 = identity) on the basis of the maximum possible score (that is, the computed score for a reflextive comparison). The normalized score, which is a measure of similarity, is returned along with the similarity list, which is an explanation of the resolution of the analogy. The user may interpret the normalized score as a measure of the closeness of the similarity among objects.

For this example, the similarity list is as follows:

[⟨ ATR_*x* & VAL_*x* (numeric) . . . . |
no occurrence
ATR_*x* & VAL_*x* (nonnumeric) ⟩ . . . .
no match

⟨ REL_*x* & OBJ_*x* ⟩	
has-cost-center 306	weight = 0.8
⟨ SET_*x* ⟩	
information management group	weight = 1.5
⟨ REL_*x* ⟩	
has-cost-center	weight = 0.6
⟨ ATR_*x* ⟩	
project-type	weight = 0.5
accounting-code	weight = 0.5
	raw score 3.9
	normalized score 6.6

The normalized score is determined by scaling the actual score to between 0 and 10 on the basis of the maximum score, which for this example is 5.9 ($3.9/5.9 = x/10$). Table 1 is a representation of this comparison.

All other algorithms build modularly on this one; for group comparisons, a common reference structure (with proportional weighting) is built for each group, and then binary comparisons are made (as above) among reference structures. Proportional weighting is necessary to maintain analogic symmetry ($A::?\ B = B::?\ A$). Numeric attributes in groups are matched within ± 1 standard deviation.

The second example is for group-weighted comparison, that is, $[A, B]::?\ [W, X, Y]$ with two objects (OBJA and OBJB) in the first group and three objects (OBJW, OBJX, and OBJY) in the second group. The similarity list for the first group (OBJA and OBJB) has already been shown. A similarity list is derived from the reference structures of the objects in the second group by the same means as in the first case. If identity of a particular element across all three objects is too strict a criterion for inclusion in the similarity list, some rule (or set of rules) can be used to decide what elements to include. Such a rule might be that more than three-quarters or one-half of the elements must be identical across objects in a group in order to be included in the similarity list for that group. Once the similarity list for the second group is derived, a binary comparison is made between group similarity lists. A normalized similarity score along with a common similarity list for both groups is returned. Tables 2 and 3 are a

Table 1. Representation of Binary-Weighted Comparison

	OBJA	OBJB	SIM[A, B]	SCORE	MAX
Attr/Value					
Proj-Type	ADMIN	RES			1.0
Acct-Code	OVD	RES			1.0
Rel/Entity					
Has-Cost-Cent	306	306	306	.8	.8
Sets					
Info-Man-GRP	Y	Y	Y	1.5	1.5
Relationships					
Has-Cost-Cent	Y	Y	Y	.6	.6
Attributes					
Proj-Type	Y	Y	Y	.5	.5
Acct-Code	Y	Y	Y	.5	.5
Total				3.9	5.9
Normalized score =					6.6

Table 2. Similarity in Group *W*, *X*, *Y*

	OBJ*W*	OBJX	OBJY	SIM[W, X, Y]	SCORE
Attr/Values					
Proj-Type	ADMIN	RES	FACL		
Acct-Code	OVD	RES	OVD	OVD	1.0 * 2/3 = .67
Manager			JOHNSON		
Rel/Entity					
Has-Cost-Cent	291	306	306	306	.8 * 2/3 = .53
Sets					
Info-Man-GRP	N	Y	Y	Y	1.5 * 2/3 = 1.0
Engineering	Y	Y	Y	Y	1.5
Relationships					
Has-Cost-Cent	Y	Y	Y	Y	.6
Attributes					
Proj-Type	Y	Y	Y	Y	.5
Acct-Code	Y	Y	Y	Y	.5
Manager	N	N	Y	N	

Table 3. Binary-Weighted Comparison of Groups [*A*, *B*] and [*W*, *X*, *Y*]

	SIM [A, B]	SCORE [A, B]	SIM [W, X, Y]	SCORE [W, X, Y]	SCORE
Rel/Entity					
Has-Cost-Cent	306	.8	306	.53	.53
Sets					
Info-Man-GRP	Y	1.5	Y	1.0	1.0
Relationships					
Has-Cost-Cent	Y	.6	Y	.6	.6
Attributes					
Proj-Type	Y	.5	Y	.5	.5
Acct-Code	Y	.5	Y	.5	.5
Total					3.13
Normalized score =					8.02

representation of this group comparison. A cutoff value of 1/2 has been used to determine the similarity lists for each group. Scoring is done by taking the minimum possible value for matches at each level of the similarity lists of the groups.

The open analogy (A::?) is computed by applying each element of the reference structure of the given object as a query in the knowledge base and building and merging similarity lists on the basis of the returns of these queries. If Object$A$ from the previous example is taken, then a query would be made to determine what other objects had "project-type administrative." The result of this query, a set of instances along with the appropriate weights, would be kept as a list by object. Then a query would be made to determine what other objects had "accounting-code adm." The returned set of objects from this query along with the appropriate weights would be merged by object with the previous list. This would be continued until queries had been made for each element of the reference structure of Object$A$. Normalized scores for each object on the merged return list would be calculated, and those objects scoring above a certain level (user specified) would be returned along with their similarity lists as the resolution of the analogy.

DISCUSSION

Conventional information management systems can be thought of as providing a highly specific form of analogy. That is, users can fully specify the descriptive characteristics of the objects they are interested in, and the system will return only those objects which meet the description. Requests in a database system are generally of this form. The relational pseudoquery

```
for c in colleges
    return c.college_name and c.town
            where c.state = "NH"
end_for
```

will return a list of colleges in New Hampshire along with the towns they are located in. The description here is fully specified.

As stated earlier, knowledge management systems must provide the user with capabilities for nontrivial inference. In the fully specified case above, the inference made is trivial. The form of computational analogy described here, and particularly the open analogy (A::?), provides deep model-based inference with respect to the reference structure of object instances. The problems addressed are of the type described by Polya [16] for mathematical analogy. The data model approach integrates knowledge representation and analogy by providing object types, each of which has a similar reference structure. Similarity is then computed by iterative comparison of reference structures for

various cases (weighted and unweighted) such as binary comparisons, group comparisons, and open comparisons. The type of analogy described in most psychological literature could be provided by extension of these algorithms.

APPENDIX

Object Integrity

OBJ_x ::=
[: ⟨ ENTITY (ENT) ⟩ |
⟨ ATTRIBUTE (ATR) ⟩ |
⟨ RELATIONSHIP (REL) ⟩ |
⟨ SET (SET) ⟩ :]

Reference Expression *Rs*

Rs ⟨ OBJ_x ⟩ ::= ⟨ property_list ⟩

⟨ property_list ⟩ ::=

[⟨ ATR_x & VAL_x (numeric) . . . . |
ATR_x & VAL_x (nonnumeric) ⟩ . . . .
⟨ REL_x & OBJ_x ⟩ . . . .
⟨ SET_x ⟩ . . . .
⟨ REL_x ⟩ . . . .
⟨ ATR_x ⟩ . . . .]

Simple Comparison Algorithm *A*0

sim ⟨ OBJx ⟩⟨ OBJy ⟩ - Is object x similar to object y?

*A*0 ::=

1. *Rs* ⟨ OBJ_x ⟩ (*Rs*_x)
2. *Rs* ⟨ OBJ_y ⟩ (*Rs*_y)
3. Repeat
   a. For each *Rs*_x_element_1 (,..., n),
      compare *Rs*_y_element_1 (,...,n)
   b. If *Rs*_x_element_x (,...,n) = *Rs*_y_element_x (,...,n)
      then write element_x to E*A*0
   Until element_n
4. return E*A*0

Simple Quantitative Comparison Algorithm $A1$

sim ($\rangle P$) ⟨ OBJx ⟩ ⟨ OBJy ⟩ - Is object x $\rangle P$ similar to object y?

$A1$ ::=

1. Rw ⟨ OBJ_x ⟩ (Rw_x)

   Rw ::= ⟨ WEIGHT_LIST ⟩

   ⟨ WEIGHT_LIST ⟩ ::= ⟨ property_list ⟩ ⟨ weight ⟩

   [⟨ ATR_x & VAL_x (numeric, 1.0) . . . . |
   ATR_x & VAL_x (nonnumeric, 1.0) ⟩ . . . .
   ⟨ REL_x & OBJ_x (0.8) ⟩ . . . .
   ⟨ SET_x (1.5) ⟩ . . . .
   ⟨ REL_x (0.6) ⟩ . . . .
   ⟨ ATR_x (0.5) ⟩ . . . .]

2. Rw ⟨ OBJ_y ⟩ (Rw_y)
3. Repeat
   a. For each Rw_x_element_1 (,...,n),
      compare Rw_y_element_1 (,...,n)
   b. If Rw_x_element_x (,...,n) = Rw_y_element_x (,...,n)
      then sum ⟨ weight ⟩ element_x and write element_x and weight
      sum for all elements to E$A1$

   Until element_n
4. If weight sum E$A1$ ⟩ P then return true else return false
5. return E$A1$

Group Quantitative Comparison Algorithm $A2$

sim ($\rangle P$)⟨OBJx,.....,OBJm⟩ ⟨OBJy,.....,OBJn⟩ – Is object x $\rangle P$
similar to the minimal similarity characteristic of objects x, . . . , n?

$A2$ ::=

1. Rc ⟨ OBJ_x ⟩ (Rc_x)

   Rc ::= ⟨ STAT_WEIGHT_LIST ⟩

   ⟨ STAT_WEIGHT_LIST ⟩ ::=

   ⟨ property_list ⟩ ⟨ weight ⟩ ⟨ stats ⟩

   [⟨ ATR_x, . . . , j & VAL_x, . . . j (numeric, 1.0) |

   ⟨ stats ⟩ ::=

   Repeat
   1. sum_a ⟨ ATR_x, . . . , j & VAL_x, . . . , j ⟩
   2. count_a –⟩ count x, . . . , j

3. *m_a* −⟩ sum_*a* / count_*a*
4. sum_*p* −⟩ (VAL_*x*, . . . , *j* * VAL_*x*, . . . *j*)
5. *v_a* −⟩ (sum_*p* −
   ((sum_*a***2)/count_*a*)/(count_*a* − 1))
6. *d_a* −⟩ sqr(*v_a*)
7. *l_m* −⟩ *m_a* − *d_a*
8. *l_x* −⟩ *m_a* + *d_a*

Until ATR_*j*

Repeat
⟨ ATR_*j* + 1, . . . *m u* VAL_*j* + 1, . . . *m*
(nonnumeric, 1.0) ⟩ . . .
⟨ REL_*j* + 1, . . . *m u* OBJ_*j* + 1, . . . *m* (0.8) ⟩ .
. . .
⟨ SET_*j* + 1, . . . *m* (1.5) ⟩ . . . .
⟨ REL_*j* + 1, . . . *m* (0.6) ⟩ . . . .
⟨ ATR_*j* + 1, . . . *m* (0.5) ⟩ . . . .]
Until element_*m*

2. Write *m_ax*, *l_mx*, *l_xx* to *Rc_x*
3. Repeat
   a. Compare OBJ_*x*_elements_*x*, . . . *m*

      elements_*x*, . . . *m* ::=
      ⟨ ATR_*x*, . . . *m* & VAL_*x*, . . . *m* (nonnumeric) ⟩
      ⟨ REL_*x*, . . . *m* & OBJ_*y*, . . . *m* ⟩ . . . .
      ⟨ SET_*x*, . . . *m* ⟩ . . . .
      ⟨ REL_*x*, . . . *m* ⟩ . . . .
      ⟨ ATR_*x*, . . . *m* ⟩ . . . .

   b. If OBJ_*x*_element_*x* (,..., *m*) = OBJ_*x*_element_*x* + 1 (,... *m*) write OBJ_*x*_element to *Rc_x*

   Until OBJ_*x*_element_*m*
4. *Rc* ⟨ OBJ_*y* ⟩ (*Rw_y*)
5. Repeat
   a. Compare *m_ax*_ATR_1 (, ..., *j*; numeric), *m_ay*_ATR_1 (, ..., *j*; numeric)
   b. If *l_m*⟨ *m_ax*_ATR_1 ⟩ *l_x* then sum ⟨ weight ⟩ and write weight sum and *m_ax*_ATR_1 to E*A*2

   Until *m_ax*_ATR_*j*
6. Repeat
   a. For each *Rc_x*_element_*j* + 1 (, ... , *m*), compare *Rc_y*_element_*j* + 1 (, ..., *n*)
   b. If *Rc_x*_element_*j* + 1 (, ... , *m*) = *Rc_y*_element_*j* + 1

(, ... , n) then sum ⟨ weight ⟩ element_x and write element_x and weight sum for all elements to E$A2$

Until element_n

7. If weight sum E$A2$ ⟩ P then return true else return false
8. return E$A2$

Unspecified Quantitative Similarity Algorithm $A3$

sim ()P)⟨ OBJx ⟩ – What objects are ⟩P similar to object x?

$A3$ ::=

1. Rw ⟨ OBJ_x ⟩ (Rw_x)
2. Repeat
   a. for each Rw_x_ATR_y, ... , n & VAL_y, ... , n (numeric)
   b. FIND OBJ_k, ... l with ATR_y, ... ,n & VAL_y, ... n (numeric)
   c. write OBJ_k, ... , l and weights to temp4a

   Until Rw_x_ATR_n & VAL_n
3. Repeat
   a. for each Rw_x_ATR_y, ... , n & VAL_y, ..., n (nonnumeric)
   b. FIND OBJ_k, ... l with ATR_y, ... , n & VAL y, ... n (nonnumeric)
   c. write OBJ_k, ... , l and weights to temp4b

   Until Rw_x_ATR_n & VAL_n
4. Union temp4a & temp4b = E$A4$ and sum weights for each OBJ_k, ... , l in E$A4$
5. Repeat
   a. for each Rw_x_REL_y, ... , n & OBJ_y, ... , n
   b. FIND OBJ_k, ... l with REL_y, ... , n & OBJ_y, ... n
   c. write OBJ_k, ... , l and weights to temp4c

   Until Rw_x_REL_n & OBJ_n
6. Union temp4c & E$A4$ = E$A4$ and sum weights for each OBJ_k, ... , l in E$A4$
7. Repeat
   a. for each Rw_x_SET_y, ... , n
   b. FIND OBJ_k, ... l with SET_y, ... , n
   c. write OBJ_k, ... , l and weights to temp4d

   Until Rw_x_SET_n

8. Union temp4d & E$A4$ = E$A4$ and sum weights for each OBJ_k, ... , l in E$A4$
9. Repeat
   a. for each Rw_x_REL_y, ... , n
   b. FIND OBJ_k, ... l with REL_y, ... , n
   c. write OBJ_k, ... , l and weights to temp4e
   Until Rw_x_REL_n
10. Union temp4e & E$A4$ = E$A4$ and sum weights for each OBJ_k, ... , n in E$A4$
11. Repeat
   a. for each Rw_x_ATR_y, ... , n
   b. FIND OBJ_k, ... l with ATR_y, ... , n
   c. write OBJ_k, ... , l and weights to temp4f
   Until Rw_x_ATR_n
12. Union temp4f & E$A4$ = E$A4$ and sum weights for each OBJ_k, ... , l in E$A4$
13. Return each OBJ_k, ... , l in E$A4$ where weight sum $\rangle P$

References

1. Hartzband, D. J., The anatomy of knowledge base management, *Proceedings of the 1st Annual Artificial Intelligence and Advanced Computer Technology Conference,* Long Beach, Calif., 333-340, 1985.
2. Hartzband, D. J. and Maryanski, F. J., Knowledge representation in engineering databases, *IEEE Computer* 18(9), 39-48, 1985.
3. Guralnik, D. B. (Ed.) *Webster's New World Dictionary of the American Language,* Popular Library, New York, 1985.
4. Spearman, C. *The Nature of Intelligence and the Principles of Cognition,* Macmillan, London, 1923.
5. Spearman, C. *The Abilities of Man,* Macmillan, New York, 1927.
6. Sternberg, R. J. *Intelligence, Information Processing and Analogic Reasoning,* Hillsdale, N.J., 1977.
7. Shalom, H. and Schlesinger, I. M., Analogical thinking: A conceptual analysis of analogy tests, in *Studies in Cognitive Modifiability,* Report 1, Vol. II (R. Fuerstein, I. M. Schlesinger, H. Shalom, and H. Narrol, Eds.), Hadassah Wizo Canada Research Institute, Jerusalem, Israel, 260-313, 1972.
8. Rummelhart, D. E. and Abrahamson, A. A., A model for analogic reasoning, *Cog. Psych.* 5, 1-28, 1973.

9. Reitman, W. *Cognition and Thought,* John Wiley, New York, 1965.

10. Evans, T. G., A program for the solution of geometric-analogy intelligence test questions, in *Semantic Information Processing* (M. Minsky, Ed.), MIT Press, Cambridge, Mass., 271–353, 1968.

11. Winston, P. H., Learning Structural Descriptions from Examples, AI Lab. Tech. Report AI TR-231, MIT, Cambridge, Mass., 1970.

12. Williams, D. S., Computer program organization induced from program examples, in *Representation and Meaning: Experiments with Information Processing Systems* (H. A. Simon and L. Siklossy, Eds.), Prentice-Hall, Englewood Cliffs, N.J., 143–205, 1972.

13. Carbonell, J. G., Learning by analogy: Formulating and generalizing plans from past experience, in *Machine Learning, An Artificial Intelligence Approach* (R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds.), Tioga Press, Palo Alto, Calif., 137–161, 1983.

14. Carbonell, J. G., Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition, CMU-CS-85-115, Computer Science Department, Carnegie-Mellon Univ., Pittsburgh, Penn., 1985.

15. Silverman, B. G., Analogy in systems management: A theoretical inquiry, *Systems, Man, Cybernetics IEEE Trans.* 13(6), 1049–1075, 1983.

16. Polya, G., *Induction and Analogy in Mathematics. Mathematics and Plausible Reasoning,* Vol. 1, Princeton U.P., Princeton, N.J., 1954.

17. Lenat, D. B. and Brown, J. S., Why AM and EURISKO appear to work, *AI* 23(3), 269–294, 1984.

18. Tversky, A., Features of similarity, *Psych. Rev.* 84, 327–352, 1977.